BIVALENT QUADRATIC PROGRAMMING PROBLEM
A COMPUTATIONAL STUDY

by

KALATHURU PRAKASAM REDDY

INDUSTRIAL AND MANAGEMENT ENGINEERING PROGRAMME

INDIAN INSTITUTE OF TECHNOLOGY KANPUR

JUNE, 1982

BIVALENT QUADRATIC PROGRAMMING PROBLEM
A COMPUTATIONAL STUDY

A Thesis Submitted
in Partial Fulfilment of the Requirements
for the Degree of
MASTER OF TECHNOLOGY

by

KALATHURU PRAKASAM REDDY

to the

INDUSTRIAL AND MANAGEMENT ENGINEERING PROGRAMME
INDIAN INSTITUTE OF TECHNOLOGY KANPUR

JUNE, 1982

CERTIFICATE

This is to certify that the work embodied in the thesis "Bivalent Quadratic Programming Problem - A Computational Study", by Kalathuru Prakasam Reddy, has been carried out under our supervision and has not been submitted elsewhere for a degree.

(S. K. Gupta)
Assistant Professor
Mathematics Department
Indian Inst. of Tech.
Kanpur 208016

(A. K. Mittal)
Assistant Professor
Industrial and Management En
Indian Institute of Technolo
Kanpur 208016

June, 1982

ACKNOWLEDGEMENTS

I am extremely grateful to my supervisors, Dr. A.K. Mittal and Dr. S.K. Gupta for their excellent guidance and their constant encouragement and inspiration.

I take this opportunity to thank all my friends, specially Naidu, Ramana, Ramekar and Gopi who made my stay at I.I.T. Kanpur a pleasant one.

I would also like to express my gratitude to Mr. J.K. Misra for his excellent typing of the thesis.

(K. P. Reddy)

June, 1982

CONTENTS

ABSTRACT

Branch and bound and heuristic algorithms are proposed to minimise a quadratic bivalent function. Effective pruning methods are proposed. Extensive computational results are reported for both branch and bound and heuristic algorithms.

CHAPTER I

INTRODUCTION

1.1 Introduction:

In this thesis quadratic bivalent programming problem of the following type is considered.

P: Minimize:

$$f(X) = C^T X + X^T M X$$

$$\text{Subject to } AX = b$$

$$X \in B_2^n$$

where C is a n x 1 real vector, M is a n x n real symmetric matrix, A is a m x n matrix, b is n x 1 real vector and $B_2 = \{0, 1\}$.

Several problems in graph theory, operations research, combinatorial mathematics, electrical engineering etc. can be formulated as problem P. Some such problems are: security investment in capital budgeting [12,14,15], facility layout and location [2], computer aided layout design [13], arrangement in printed circuits, arrangement of printed circuits on the computer back plane, planning of presidential election campaign [16], scheduling parallel machines with change-over costs [4], portfolio selection [17] etc.

Although the problem has such wide applications, no computationally efficient algorithms are known. It is well known that the problem is NP-hard.

Even if the objective function of P is assumed to be linear, the problem still remains difficult to solve. The essential difficulty pertains to the absence of necessary and sufficient optimality conditions for such problems. The solution procedures which have been reported in literature make use of the principles of exhaustive intelligent enumeration and relaxation etc. Such approaches usually require extremely large computational time even for small problems.

Hammer and Rudeanu [7] have proved the following results for the problem

H1: Minimize:

$$f(X)$$

Subject to $f_j(X) = 0,\ j = 1,2,\ldots,\ m$

where each of $f_j(X)$ is an integer valued pseudo-boolean positive function i.e. $f_j(X) \geq 0$ for $X \in B_2^n$ and $f(X)$ is a polynomial function.

i) If the vector $X^* = (x_1^*, x_2^*, \ldots, x_n^*)$ is a solution to the problem H1, then X^* also minimizes the pseudo-boolean function

$$F(X) = f(X) + (S^+ - S^- + 1) \sum_{j=1}^{m} f_j(X) \qquad (1.1.1)$$

where S^+ and S^- are the sum of positive and negative coefficients of f respectively.

ii) If X^* minimizes F and

$$F(X^*) \leq S^+$$

then X^* is a solution to the problem H1.

iii) If X^* minimizes F and

$$F(X^*) > S^+$$

then the constraints of the problem H1 are inconsistent.

If any of the constraints $f_j(X)$ for $j = 1,2,\ldots,m$ is not a positive function, then we replace the constraint $f_j(X) = 0$ by $(f_j(X))^2 = 0$. Since the function $(f_j(X))^2$ is a positive function, we can use equation (1.1.1) to eliminate the constraints.

Moreover, any integer valued inequality constraints of the type $f_j(X) \geq 0$ can be converted to an equivalent equality constraint as given below:

$$f_j(X) - \sum_{k=0}^{P_j} 2^k y_{k+1,j} = 0$$

with $y_{k+1,j} \in B_2$ and $2^{P_j+1} > M_j$, where M_j is an upper bound on $f_j(X)$.

Using the above results, any quadratic bivalent programming problem with linear constraints can be converted to equivalent unconstrained quadratic bivalent programming problem. Hence problem P can be converted to equivalent unconstrained quadratic bivalent programming problem.

In view of Hammer and Rudeanu's result given above, we consider the following problem for developing the solution procedure, which can be used to solve P also.

P1: Minimize:

$$f(X) = D^T X + X^T RX \qquad (1.1.2)$$

Subject to $X \varepsilon B_2^n$

where D is a n x 1 real vector, R is a n x n real symmetric matrix.

Since $x_i = x_i^2$ for $x_i = 0, 1$ linear terms in the objective function of (1.1.2) can be implicitly taken into quadratic part. Hence problem P1 can be written as,

P2: Minimize:

$$f(X) = X\,QX$$

Subject to $X \varepsilon B_2^n$

where Q is a n x n real symmetric matrix.

Rosenberg [22] has shown that any unconstrained polynomial bivalent program " minimize f(X) subject to $X \varepsilon B_2^n$ " is equivalent to an unconstrained quadratic bivalent program.

The existing solution procedures for the problem P are mainly of three kinds: i) Boolean approach, ii) Branch and Bound methods and iii) Network approach. The Boolean approach was suggested by Hammer and Rudeanu [7]. This approach does not seem to be very suitable for computation

on computer. Branch and bound methods are discussed by Hammer and other authors in [3, 8, 9, 10, 23, 24].

Hammer and Peled [9] observed that it takes 110 sec. (including input/output time) on IBM 360/50 to solve 20 variable polynomial programming problems with 50 terms in the polynomial. They further state that 30 variable polynomial problems with 70 terms in polynomial could not be solved due to memory limitation.

A number of unconstrained quadratic bivalent programs can be solved by network flow algorithms. Study of this class of problems is done by Picard and Ratliff [18-20]. But this approach has certain restrictions. For example to use algorithm given in [18] for problem P2 all the off-diagonal elements of matrix Q are required to be positive.

Methods based on implicit enumeration include the one given by Laughhun[14]. Taha [25, 26] has given an algorithm for bivalent polynomial programming problems based on Balas [1] additive algorithm.

Recently Gulati [5] proposed a branch and pruning algorithm to solve unconstrained quadratic bivalent programming problems. Using this algorithm,problems involving 125 variables could be solved within 198 seconds on DEC - 10 system.

From the above discussion, it is clear that no computationally efficient algorithm is known to solve P2.

In this thesis, we have made an attempt to develope a computationally attractive algorithm to solve P2.

1.2 Solution Approach:

A branch and bound method is developed to solve P2. Bounding strategies are developed that are found to be effective. In addition, Gulati's [5] pruning test is used, which ensures that there does not exist a local solution along a particular branch.

Branch and bound method is essentially used to identify the most difficult problems and to test the effectiveness of heuristics. To solve difficult and/or large size problems heuristics are developed.

In Chapter II, we discuss in detail the solution procedure. We discuss some heuristics in Chapter III. Chapter IV deals with computational results pertaining to the effectiveness of the algorithms proposed in this thesis.

CHAPTER II

BRANCH AND BOUND METHOD

In this chapter, we first develope certain mathematical relationships which are used in the proposed branch and bound method. The method is then described.

2.1 Neighbouring Point:

Now we define neighbouring point.

Definition 2.1.1

For any given $X = (x_1, x_2, \ldots, x_{k-1}, x_k, x_{k+1}, \ldots, x_n)$, we define,

$$X^k = (x_1, x_2, \ldots, x_{k-1}, 1-x_k, x_{k+1}, \ldots, x_n) \quad (2.1.1)$$

as a neighbouring point of X for $k = 1,2,\ldots, n$.

Now we develope certain mathematical relationships useful in the proposed method.

2.2 Mathematical Relationships:

For any given point $X \in B_2^n$,

let $I_0(X) = \{i | x_i = 0\}$

$I_1(X) = \{i | x_i = 1\}$.

Given any point $X \in B_2^n$ and the objective function value $f(X)$, the objective function value $f(X^k)$ corresponding

to a neighbouring point X^k of X for k = 1,2,..., n can be found as,

$$f(X^k) = \begin{cases} f(X) + q_{kk} + 2 \sum_{i\varepsilon I_1(X)} q_{ik} & \text{if } k \varepsilon I_o(X) \quad (2.2.1) \\ f(X) - [q_{kk} + 2 \sum_{i\varepsilon I_1(X)-\{k\}} q_{ik}] & \text{if } k \varepsilon I_1(X) \quad (2.2.2) \end{cases}$$

By combining equations (2.2.1) and (2.2.2), we can write,

$$f(X^k) = f(X) + QDIAG_k(X) \quad \text{for } k = 1,2,\ldots,n \quad (2.2.3)$$

where $QDIAG_k(X)$ is the k-th element of 1 x n size array QDIAG (X) corresponding to point X and defined as,

$$QDIAG_k(X) = \begin{cases} q_{kk} + 2 \sum_{i\varepsilon I_1(X)} q_{ik} & \text{if } k \varepsilon I_o(X) \quad (2.2.4) \\ -[q_{kk} + 2 \sum_{i\varepsilon I_1(X)-\{k\}} q_{ik}] & \text{if } k \varepsilon I_1(X) \quad (2.2.5) \end{cases}$$

In theorem (2.1), we obtain the elements of QDIAG (X^k) corresponding to a neighbouring point X^k of X for k = 1,2,..., n, in terms of QDIAG (X).

Theorem 2.1:

Given $X = (x_1, x_2, \ldots, x_n) \varepsilon B_2^n$,

for i = 1,2,..., n and k = 1,2,..., n

$$QDIAG_i(X^k) = \begin{cases} -QDIAG_i(X) & \text{if } i = k \quad (2.2.6) \\ QDIAG_i(X) + 2q_{ik} & \text{if } x_i = x_k \text{ and } i \neq k \quad (2.2.7) \\ QDIAG_i(X) - 2q_{ik} & \text{if } x_i \neq x_k \text{ and } i \neq k \quad (2.2.8) \end{cases}$$

Proof:

Case 1: If $k \in I_0(X)$ then $k \in I_1(X^k)$.

Also,

$$I_0(X^k) = I_0(X) - \{k\}$$
$$I_1(X^k) = I_1(X) \cup \{k\}$$

i) If $i = k$ then $i \in I_0(X)$, $i \in I_1(X^k)$ and

$$QDIAG_i(X^k) = -[q_{ii} + 2 \sum_{j \in I_1(X^k)-\{i\}} q_{ij}] \quad \text{(by (2.2.5))}$$

$$= -[q_{ii} + 2 \sum_{j \in I_1(X) \cup \{k\}-\{i\}} q_{ij}]$$

Since $i = k$,

$$QDIAG_i(X) = [q_{ii} + 2 \sum_{j \in I_1(X)} q_{ij}] \quad \text{(by (2.2.4))}$$

Therefore,

$$QDIAG_i(X^k) = -QDIAG_i(X).$$

ii) If $x_i = x_k$ and $i \neq k$ then

$i \in I_0(X)$, $i \in I_0(X^k)$ and

$$QDIAG_i(X^k) = q_{ii} + 2 \sum_{j \in I_1(X^k)} q_{ij} \quad \text{(by (2.2.4))}$$

$$= q_{ii} + 2 \sum_{j \in I_1(X) \cup k} q_{ij}$$

$$= q_{ii} + 2 \sum_{j \in I_1(X)} q_{ij} + 2q_{ik}$$

Since $i \in I_0(X)$

$$QDIAG_i(X) = q_{ii} + 2 \sum_{j \in I_1(X)} q_{ij} \quad \text{(by (2.2.4))}$$

Therefore,

$$QDIAG_i(X^k) = QDIAG_i(X) + 2q_{ik}.$$

iii) If $x_i \neq x_k$ and $i \neq k$ then

$i \in I_1(X)$, $i \in I_1(X^k)$ and

$$QDIAG_i(X^k) = -[q_{ii} + 2 \sum_{j \in I_1(X^k)-\{i\}} q_{ij}] \text{ (by (2.2.5))}$$

$$= -[q_{ii} + 2 \sum_{j \in I_1(x) \cup \{k\}-\{i\}} q_{ij}]$$

$$= -[q_{ii} + 2 \sum_{j \in I_1(X)-\{i\}} q_{ij} + 2q_{ik}]$$

Since $i \in I_1(X)$,

$$QDIAG_i(X) = -[q_{ii} + 2 \sum_{j \in I_1(X)-\{i\}} q_{ij}] \quad \text{(by (2.2.5))}$$

Therefore,

$$QDIAG_i(X^k) = QDIAG_i(X) - 2q_{ik}$$

<u>Case 2</u>: If $k \in I_1(X)$ then $k \in I_o(X^k)$.

Also, $I_o(X^k) = I_o(X) \cup \{k\}$

$I_1(X^k) = I_1(X) - \{k\}$

i) If $i = k$ then $i \in I_1(X)$, $i \in I_o(X^k)$ and

$$QDIAG_i(X^k) = [q_{ii} + 2 \sum_{j \in I_1(X^k)} q_{ij}] \quad \text{(by (2.2.4))}$$

$$= [q_{ii} + 2 \sum_{j \in I_1(X)-\{k\}} q_{ij}]$$

Since $i = k$,

$$QDIAG_i(X) = -[q_{ii} + 2 \sum_{j\varepsilon I_1(X)-\{k\}} q_{ij}] \quad \text{(by (2.2.5))}$$

Therefore,

$$QDIAG_i(X^k) = - QDIAG_i(X)$$

ii) If $i \neq k$ and $x_i = x_k$ then

$$i \varepsilon I_1(X), \quad i \varepsilon I_1(X^k) \text{ and}$$

$$QDIAG_i(X^k) = - [q_{ii} + 2 \sum_{j\varepsilon I_1(X^k)-\{i\}} q_{ij}] \text{ (by(2.2.5))}$$

$$= -[q_{ii} + 2 \sum_{j\varepsilon I_1(X)-\{k\}-\{i\}} q_{ij}]$$

$$= -[q_{ii} + 2 \sum_{j\varepsilon I_1(X)-\{i\}} q_{ij}] + 2q_{ik}$$

Since $i \varepsilon I_1(X)$,

$$QDIAG_i(X) = - [q_{ii} + 2 \sum_{j\varepsilon I_1(X)-\{i\}} q_{ij}] \quad \text{(by(2.2.5))}$$

Therefore,

$$QDIAG_i(X^k) = QDIAG_i(X) + 2q_{ik}$$

iii) If $i \neq k$ and $x_i \neq x_k$ then

$$i \varepsilon I_o(X), \quad i \varepsilon I_o(X^k) \text{ and}$$

$$QDIAG_i(X^k) = [q_{ii} + 2 \sum_{j\varepsilon I_1(X^k)} q_{ij}] \qquad \text{(by(2.2.4))}$$

$$= [q_{ii} + 2 \sum_{j\varepsilon I_1(X)-\{k\}} q_{ij}]$$

$$= [q_{ii} + 2 \sum_{j\varepsilon I_1(X)} q_{ij}] - 2q_{ik}$$

Since $i \varepsilon I_o(X)$,

$$QDIAG_i(X) = [q_{ii} + 2 \sum_{j \varepsilon I_1(X)} q_{ij}] \quad \text{(by (2.2.4))}$$

Therefore,

$$QDIAG_i(X^k) = QDIAG_i(X) - 2q_{ik}$$

This completes the proof.

2.3 Local Minimization:

In this section we introduce the concept of locally minimizing point of P2.

2.3.1 Locally Minimizing Point:

Definition 2.3.1: $X \varepsilon B_2^n$ is said to be a locally minimizing point of P2 if,

$$f(X) \leq f(X^k) \text{ for } k = 1,2,\ldots, n$$

where X^k is defined in (2.1.1).

Theorem 2.2:

$X \varepsilon B_2^n$ is a locally minimizing point of P2 if and only if,

$$QDIAG\ (X) \geq 0 \qquad (2.3.1)$$

Proof:

From (2.2.3), we have,

$$f(X^k) = f(X) + QDIAG_k(X) \text{ for } k = 1,2,\ldots, n \quad (2.3.2)$$

Let $X \varepsilon B_2^n$ be a locally minimizing point of P2, then

$$f(X) \leq f(X^k) \text{ for } k = 1,2,\ldots, n.$$

From (2.3.2), we get,

$QDIAG_k(X) \geq 0$ for $k = 1,2,\ldots, n$.

Hence X, a locally minimizing point satisfies (2.3.1).

Conversely, let $X \varepsilon B_2^n$ satisfy (2.3.1), then from (2.3.2), we get,

$f(X^k) \geq f(X)$ $k = 1,2,\ldots, n$

which shows that X is a locally minimizing point of P2.

This completes the proof.

We now present an algorithm to determine a locally minimizing point of P2. We start with a point $X \varepsilon B_2^n$.

2.3.2 Algorithm 2.1: LOCAL

Step 1: Select an initial point $X \varepsilon B_2^n$ and compute,

$QDIAG_i(X)$, $i = 1,2,\ldots, n$, $f(X)$

Step 2: If $QDIAG(X) \geq 0$ go to Step 4 else choose k according to one of the rules specified below and go to Step 3.

Step 3: Compute $f(X^k) = f(X) + QDIAG_k(X)$. Compute $QDIAG_i(X^k)$, $i = 1,2,\ldots, n$ by using equations (2.2.6) to (2.2.8). Update X to X^k. Go to Step 2.

Step 4: Stop, X is a locally minimizing point of P2 and f(X) is the value of the objective function at this point.

2.3.3 Rules for Selecting k in Step 2 of Algorithm LOCAL:

Rule 1: Select the smallest k such that $QDIAG_k(X) < 0$.

Rule 2: Select the k such that

$$QDIAG_k(X) = \min_i [QDIAG_i(X)] \quad i = 1,2,\ldots, n$$

Rule 3: (LRC Rule): Let the indices of QDIAG (X) be arranged in circular fashion. Following a fixed direction, choose that k which is nearer to the index that was choosen in previous step and $QDIAG_k(X) < 0$.

The justification for the algorithm 2.1 can be argued as follows:

We start with the point X $(x_1, x_2, \ldots, x_n) \varepsilon B_2^n$ and at each iteration the objective function value decreases. We stop at a point which satisfies the necessary and sufficient conditions for a point to be locally minimizing as given in theorem 2.2. Since the objective function value decreases at every iteration and the number of points available is finite, the algorithm terminates in a finite number of iterations.

2.4 Global Minimization:

In this section, we discuss the pruning tests, and also the branch and bound method to solve P2.

2.4.1 Pruning Test:

This is same as the pruning test used by Gulati, Mittal and Gupta [6]. Now we briefly explain the test.

Let $\{U_o, U_1, U_f\}$ be any partition of $\{1,2,\ldots, n\}$

where, $U_o = \{i \mid i \text{ is fixed and } i \in I_o(X)\}$

$U_1 = \{i \mid i \text{ is fixed and } i \in I_1(X)\}$

$U_f = \{i \mid i \text{ is free }\}$

If we fix $x_i = 0$ for $i \in U_o$, $x_i = 1$ for $i \in U_1$, then the pruning test gives sufficient conditions under which there does not exist an assignment of values of the free variables x_i, $i \in U_f$, such that the corresponding completion $(x_1, x_2, \ldots, x_n)$ is a local minimum of P2.

If x_i, $i \in U_f$, is any assignment of the free variables, then the corresponding completion $X = (x_1, x_2, \ldots, x_n) \in B_2^n$, where $x_i = 0$ for $i \in U_o$ and $x_i = 1$ for $i \in U_1$. For a fixed assignment x_i^o for $i \in U_f$, of the free variables, let X^o be the corresponding completion, and for $s \in U_o \cup U_1$ let,

$$A_1 = \{i \in U_f \mid q_{si} > 0\} \cap I_1(X^o)$$

$$A_2 = \{i \in U_f \mid q_{si} < 0\} \cap I_o(X^o)$$

$$A_3 = \{i \in U_f \mid q_{si} > 0\} \cap I_o(X^o)$$

$$A_4 = \{i \in U_f \mid q_{si} < 0\} \cap I_1(X^o)$$

Theorem 2.3: If $QDIAG_s(X^o) < 0$ for some $s \in U_1$, such that

$$\sum_{i \in A_1} 2q_{si} + \sum_{i \in A_2} 2|q_{si}| < |QDIAG_s(X^o)| \qquad (2.4.1)$$

then there does not exist any assignment of free variables for which the completion is a local minimum of P2.

Proof is given in [6].

Theorem 2.4:

If $QDIAG_s\ (X^o) < 0$ for some $s \in U_o$, such that

$$\sum_{i \in \Lambda_3} 2q_{si} + \sum_{i \in \Lambda_4} 2\,|q_{si}| < |QDIAG_s\ (X^o)| \quad (2.4.2)$$

then there does not exist any assignment of the free variables for which the completion is a local minimum of P2.

Proof is given in [6].

2.4.2 Lower Bound:

In this section, we develope an effective lower bound for problem P2.

Gallo, Hammer and Simeone [3] stated that if V_j is a lower bound for $\min \{q_j^T X : X \in S\}$, where q_j is the j-th column of Q, and S is constraints set, then the function $\sum_{j=1}^{n} V_j x_j$ is obviously a lower plane for $f(X) = \sum_{j=1}^{n} (q_j^T X) x_j$ in S. Using this result, we develope a lower bound for problem P2 as given below.

Let $V_o = \sum_{i,j \in U_1} q_{ij}$

$$V_j = 2 \sum_{i \in U_1} q_{ij} + \min \{ \sum_{i \in U_f} q_{ij} x_i : x_i \in B_2 \},$$

for $j \in U_f$

Then lower bound for P2 can be written as,

$$LB = V_o + \min \{ \sum_{j \in U_f} V_j x_j : x_j \in B_2 \}$$

$$= V_o + \sum_{j \in U_f} \min [0, V_j]$$

When $V_j < 0$, if x_j is allowed to take 1, then q_{jj} will be added to objective function irrespective of the sign of q_{jj}. Hence V_j is modified to,

$$V_j' = q_{jj} + 2 \sum_{i \in U_1} q_{ij} + \sum_{i \in U_f - \{j\}} \min [0, q_{ij}],$$

for $j \in U_f$ (2.4.3)

If free variable x_j has value zero, then,

$$V_j' = QDIAG_j (X) + \sum_{i \in U_f - \{j\}} \min [0, q_{ij}], j \in U_f$$

in view of (2.2.4)

Therefore,

$$LB = V_o + \sum_{j \in U_f} \min [0, V_j'] \qquad (2.4.4)$$

Since the effectiveness of our bound is dependent on the sum of negative elements of the sub matrix corresponding to free variables, it will be benificial to rearrange the elements of matrix Q such that the sums,

$$q_{ii} + 2 \sum_{\substack{j=1 \\ j \neq i}}^{n} \min [0, q_{ij}] \quad \text{for } i = 1, 2, \ldots, n \qquad (2.4.5)$$

are in ascending order.

2.4.3 Trivial Test:

Now we give certain equations that may help us in reducing the size of matrix Q.

For any $i = 1,2,\ldots, n$, if $q_{ii} < 0$ and

$$q_{ii} + 2 \sum_{\substack{j=1 \\ j \neq i}}^{n} \max [0, q_{ij}] \leq 0 \qquad (2.4.6)$$

then x_i will take value 1 in any optimal solution. Then we modify the diagnol elements of Q as,

$$q_{jj} = q_{jj} + 2q_{ij}, \quad j = 1,2,\ldots, n, \quad j \neq i,$$

for i that satisfied (2.4.6) and delete column and row of Q corresponding to x_i.

Similarly for any $i = 1,2,\ldots, n$,

if $q_{ii} \geq 0$ and

$$q_{ii} + 2 \sum_{\substack{j=1 \\ j \neq i}}^{n} \min [0, q_{ij}] \geq 0 \qquad (2.4.7)$$

then x_i will take value 0 in any optimal solution. Then, we delete column and row of Q corresponding to x_i.

By repeatedly using equations (2.4.6) and (2.4.7) and making necessary modifications to Q, it may be possible to reduce size of matrix Q.

2.4.4 Branch and Bound Method:

The proposed method is essentially a depth first search method. We identify any node by the point $X = \{x_1, x_2, \ldots, x_n\}$ and the level upto which the variables are fixed.

Branching Strategy: Branching variables are selected in ascending order of their indices. The branch having the value of branching variable equal to zero is explored first.

Type of Node: At any node the completion X is found by setting free variables at zero value. We identify the node as type I if the completion is a locally minimizing point for P2, otherwise as type II. It can be observed that the type of successor node obtained by fixing the free variable at zero value is same as that of parent node.

Bounding Test: At any node we calculate the lower bound using (2.4.4) and if lower bound $\geq$ incumbent we prune that branch.

Pruning Strategy: At any node pruning is done using bounding test only if the type of node is I, otherwise we use both bounding test and pruning test.

A formal description of the algorithm to generate the global solution is given below.

Algorithm 2.2 GLOBAL:

Step 1: Initialize level = 0, go to step 2.

Step 2: Use algorithm LOCAL to find a local solution X. Set the optimal solution X^* as X and $f(X^*)$ as $f(X)$. Set incumbent as $f(X)$. Also set $X = \{0,0,0,\ldots 0\}$, $f(x) = 0$, Calculate QDIAG(X).

Step 3: Set level = level + 1, go to Step 4.

Step 4: If level equals to number of variables go to Step 8 else go to Step 5.

Step 5: Apply bounding test. If bounding test succeeds go to Step 8 else go to Step 6.

Step 6: If type of node is I, go to Step 3, else go to Step 7.

Step 7: Apply pruning test. If test fails, go to Step 3, else go to Step 3.

Step 8: If $x_{level} = 1$, go to Step 9, else set $x_{level} = 1$; calculate X, QDIAG (X), f(X). Update X^*, $f(X^*)$, incumbent, go to Step 4.

Step 9: If level equals to 1, go to Step 10, else set $x_{level} = 0$. Calculate X, QDIAG (X), f(X). Set level = level - 1, go to Step 8.

Step 10: Stop. $X^* = \{x_1^*, x_2^*, \ldots, x_n^*\}$ is the optimal solution. $f(X^*)$ is the objective function value.

Illustration:

Following numerical example illustrates the proposed method for n = 5 (Algorithm GLOBAL).

$$Q = \begin{bmatrix} -29 & 0 & 15 & 15 & 1 \\ 0 & -68 & 19 & 15 & 17 \\ 15 & 19 & -112 & 6 & 17 \\ 15 & 15 & 6 & 0 & 7 \\ 1 & 17 & 17 & 7 & -56 \end{bmatrix}$$

Using algorithm 2.1 (LOCAL) with starting point X = (0,0,...,0) we get the first locally minimizing point as $x_1 = 0$, $x_2 = 1$, $x_3 = 1$, $x_4 = 0$, $x_5 = 0$. Value of the objective function is -142. So incumbent INC = -142 for branch and bound method. And the values of QDIAG corresponding to this point are (1, 30, 74, 42, 12).

For branch and bound method, we start with point X = (0, 0, 0, 0, 0). First we fix x_1 at 0. Lower bound is found to be equal to -236. Type of node is II. Both pruning and bounding tests fail.

We now fix x_2 at 0. Lower bound is equal to -168. Type of node is II. Again both the tests fail. We now fix x_3 at 0. Lower bound is equal to -56. Since lower bound > INC, we prune this branch and search along $x_3 = 1$. With $x_3 = 1$ type of node is II. Lower bound is -134 > INC. So we prune this branch also.

Now we trace back and fix x_2 at 1. Now lower bound is equal to -164. Type of node is II. Both the tests fail. When we fix x_3 at 0 lower bound is equal to -90. Since lower bound > INC, we prune this branch. Now we fix x_3 at 1. Lower bound is equal to -142. Since lower bound = INC, we prune this branch also.

Now again, we trace back and fix x_1 at 1. Bounding test succeeds with $x_2 = 0$, $x_3 = 0$; $x_2 = 0$, $x_3 = 1$; $x_2 = 1$, $x_3 = 0$ and $x_2 = 1$, $x_3 = 1$.

We stop at this stage. Optimal solution is (0, 1, 1, 0, 0) and optimal value of the objective function is -142.

Complete description of algorithm is shown in Fig. (2.1).

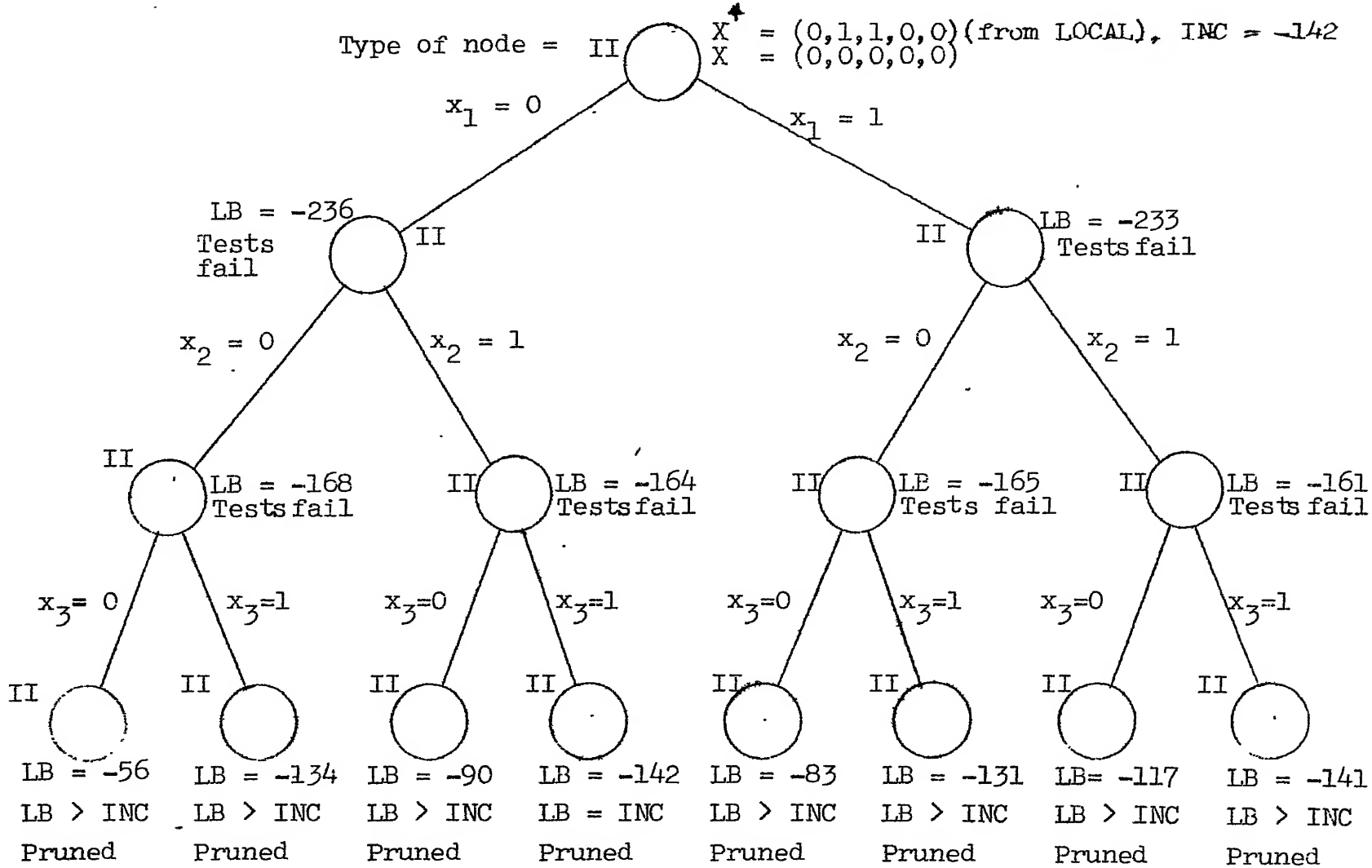

Type of node = II
X* = (0,1,1,0,0)(from LOCAL), INC = -142
X = (0,0,0,0,0)
x1 = 0
x1 = 1
LB = -236
Tests fail
II
II
LB = -233
Tests fail
x2 = 0
x2 = 1
x2 = 0
x2 = 1
II
LB = -168
Tests fail
II
LB = -164
Tests fail
II
LB = -165
Tests fail
II
LB = -161
Tests fail
x3= 0
x3=1
x3=0
x3=1
x3=0
x3=1
x3=0
x3=1
II
II
II
II
II
II
II
II
LB = -56
LB > INC
Pruned
LB = -134
LB > INC
Pruned
LB = -90
LB > INC
Pruned
LB = -142
LB = INC
Pruned
LB = -83
LB > INC
Pruned
LB = -131
LB > INC
Pruned
LB= -117
LB > INC
Pruned
LB = -141
LB > INC
Pruned

Fig. 2.1

CHAPTER III

HEURISTICS

3.1 Introduction:

In this chapter, we discuss the heuristics that can be used to solve large size problems within reasonable amount of time. Heuristics provide near-optimal solutions in general.

All the heuristics that we discuss in this chapter essentially consist of finding a local solution or a set of local solutions. We pick-up the best among the set of local solutions. These heuristics differ from each other in identifying the starting points for algorithm LOCAL. Once the starting point is identified, we use algorithm LOCAL to find local solution. This study also helps us in finding out starting point strategy to be used in algorithm LOCAL to find initial incumbent to branch and bound method.

We now give different strategies for identifying starting points and use them as heuristics to solve problem P2.

3.1.1 Starting Point Strategies:

3.1.1.1 All 0's: In this method, we set starting point as $X = \{0, 0, \ldots, 0\}$ and then use algorithm LOCAL to find local solution.

3.1.1.2 All 1's: Here we set the starting point as $X = \{1, 1, \ldots, 1\}$ and use algorithm LOCAL to find local solution.

3.1.1.3 Random Start: Similar methods are used to solve Quadratic assignment problems [21]. This method essentially consists of randomly generating set of K starting points and then finding the local solution for each of these starting points. We prefix the cardinality of 1's in the starting point and then select the variables to take value 1 using uniform random numbers. We hope that this ensures better coverage of local solutions. For our computational study we will prefix the cardinality of 1's as (.1n, .2n, .3n, ..., .9n) and hence the value of K = 9.

3.1.1.4 Constructive Method:

This method is similar to random start method except that in each of the starting point that we generate, we allow a particular set of variables to take 1's and the rest zero. As mentioned in Chapter II, if the elements of matrix Q are rearranged such that the sums,

$$q_{ii} + 2 \sum_{\substack{j=1 \\ j \neq i}}^{n} \min (0, q_{ij}), \qquad i = 1,2,\ldots, n$$

are in ascending order, then we expect that the chance of a variable taking one go down with increase in index. So we fix first few elements of X at 1. Number of variables

to take one is equal to the cardinality of 1's. For computational study, we prefix the cardinality of 1's as (.1n, .2n, ..., .9n). We use algorithm LOCAL to find local solution for each of the starting points and we pick up the best among these local solutions.

3.1.1.5 Epsilon Method:

In this method, we construct a starting point by relaxing equations (2.4.6) and (2.4.7) as given below.

For any $i = 1,2,\ldots, n$

if $q_{ii} < 0$, and

if $q_{ii} + \varepsilon \times 2 \sum_{\substack{j=1 \\ j\neq i}}^{n} \max(0, q_{ij}) \leq 0$ set $x_i = 1$

else set $x_i = 0$,

if $q_{ii} \geq 0$, and

if $q_{ii} + \varepsilon \times 2 \sum_{\substack{j=1 \\ j\neq i}}^{n} \min(0, q_{ij}) \geq 0$ set $x_i = 0$

else set $x_i = 1$

Here ε is to be judiciously prescribed. We use algorithm 2.1 to find the local solution.

CHAPTER IV

COMPUTATIONAL RESULTS AND CONCLUSIONS

4.1 Introduction:

In this chapter, we shall discuss computational performance of the algorithms developed in the previous chapters. Computations have been performed on DEC 10 (KL 40 processor). Programs are coded in FORTRAN 10 (DEC version). Actual run time is taken to be the computational time.

For the test problems, the elements of matrix Q are generated using uniformly distributed random numbers, such that the matrix Q is symmetric and the elements of Q are integers. Two types of problems are considered i.e. 1) Problems with only on-diagnol elements of matrix Q being negative, 2) Problems with negative elements distributed throughout the matrix Q.

We hypothesize that the computational performance of the algorithms is dependent on the following factors:

i) Density of matrix Q.

ii) Ratio of size of on-diagonal elements to that of off-diagonal elements of Q denoted as I.

iii) Percentage of negative elements in total non-zero elements of matrix Q, denoted as IN. (Applicable only for the problems with negative elements distributed throughout the matrix Q).

The elements q_{ij} $i \neq j$ of the matrix Q are generated from a uniform probability distribution U [0, 100] for all problems. The diagonal elements are generated by multiplication of U[-100, 0] and I for problems with only on-diagonal elements negative and by multiplication of U[0, 100] and I for problems with negative elements distributed throughout matrix Q. A random number is generated to control the density of the matrix Q. For problems with negative elements distributed throughout Q, a ratio of negative to non-zero elements is prefixed and the sign of the specific element is generated using uniform random number.

After generating matrix Q, we apply trivial test to assign variable value 1 or 0 as discussed in Section 2.4.3 and reduce the problem accordingly, if possible. Then we rearrange the matrix Q such that the sums given in (2.4.5) are in ascending order.

4.2 Computational Study and Results:

4.2.1 Computational Evaluation of Rules to Select the Variables in Algorithm LOCAL:

We evaluate the following rules for selecting variable in step 2 of algorithm LOCAL (Sec. 2.3.2) outlined in Sec. 2.3.3.

i) Rule 1: Selecting the highest negative element of QDIAG.

ii) Rule 2: Selecting the first negative element of QDIAG.

iii) Rule 3: LRC rule.

Performance is evaluated using the following criteria.

i) Number of problems, for which the objective function value obtained by using a particular rule is better than or atleast as good as the remaining two rules.

ii) Number of iterations required to find a local solution when a particular rule is used.

The study is carried out on problems with negative elements distributed throughout matrix Q. Problems of size n = 20 and n = 30 are considered. For each value of n, 30 problems are solved by varying density (.25, .50, 1.0), I (1, 3, 5, 7, 9) and IN (30, 50).

Results of this study are reported in Table 4.2.1.

It is observed that the rule 1 that chooses the variable with highest negative/QDIAG element of dominates the other two rules in both the criteria.

4.2.2 Computational Evaluation of Different Starting Point Strategies Used in Algorithm LOCAL:

We evaluate the following strategies used for identifying the starting points in algorithm LOCAL (Sec. 2.3.2)

outlined in Sec. 3.1.1.

S1: All 0's

S2: All 1's

S3: Random start: In this method we consider generation of random starting points with cardinality of 1's in X prefixed at (.1n, .2n,..., .9n) and hence K = 9 as outlined in Sec. 3.1.1.3.

S4: Constructive method: In this method also we prefix the cardinality as given above and variables to take 1 are decided as outlined in Sec. 3.1.1.4.

S5: Epsilon Method: Extensive computations have been performed by varying the value of the parameter ε (Sec. 3.1.1.5) and at $\varepsilon = 0.2$ the results are found to be encouraging. Hence we fix $\varepsilon = 0.2$ for computational study.

Performance is evaluated using the following criteria.

1. Number of problems for which the objective function value obtained by using particular strategy is better than or at least as good as the remaining strategies.

2. Number of problems for which the objective function value for a specific strategy is within 10 percent of the best solution.

3. Maximum error with respect to best solution, where error is defined as:

If f_{S_j} is the objective function value obtained by strategy S_j, then

$$\text{percentage error for strategy } S_j = \frac{(\min_i f_{S_i} - f_{S_j}) \times 100}{\min_i f_{S_i}}$$

Strategies are tested on problems 1) with only on-diagonal elements of matrix Q negative, 2) with negative elements distributed throughout matrix Q. In both the cases problems with number of variables n = 50 and 100 are considered. Other input parameters are varied as follows: Density (0.1, 0.25, 0.50), I (1, n x density/2, n x density) and IN (30, 50, 70). For each value of n, 27 problems are solved.

Results are reported in Table 4.2.2.

It is observed that the random start strategy dominates all other strategies in both the criteria.

Further to study the relationship of the cardinality of l' of random starting solution with respect to the best local solution, the following problem is considered.

number of variables n = 100

Density of matrix Q = 0.1

Percentage of negatives = 50

Number of problems solved = 20

Cardinality of starting point is varied as (.05n, .1n, ..., .95n). It is observed that 75 percent of the best local solutions are covered by the starting points with cardinality greater than .5n. So for large size problems, it may be worth to use more random starting points with higher cardinality of 1's.

Now, we suggest the following heuristic to solve large size problems.

Heuristic BEST LOCAL:

i) Select K random start points with cardinality of 1's being prefixed.

ii) For each of the starting points, get a local solution using algorithm LOCAL.

iii) Select the best local solution.

4.2.3 Computational Evaluation of Branch and Bound (GLOBAL) and Heuristic BEST LOCAL:

We evaluate the effectiveness of heuristic in terms of the computational time reqd. and the average percentage error.

Percentage error is calculated as given below:

$$\text{Percentage error} = \frac{(\text{Optimal solution value} - \text{heuristic solution value})}{\text{Optimal solution value}} \times 100$$

We also identify the most difficult problems to solve.

Study is carried out on two types of problems:

i) Only on diagonal elements of Q are negative: We solve 5 problems for each of the following combinations.

n (25, 30), Density (0.25, 0.50, 0.75, 1.00),

I (1, 3, 5, ..., 19)

For n = 30 and density = 0.75, 1, in addition to the above combinations we solve problems with

I = (21, 23, ..., 29) also.

Results are shown in Tables (4.2.3) to (4.2.7).

ii) Negative elements of Q are distributed throughout the matrix:

We solve 5 problems for each of the following combinations.

n (25, 30), Density (0.25, 0.5, 0.75, 1.00),

I (1, 6, 11) and IN (10, 30,50, 70).

Results are reported in Tables (4.2.8) to (4.2.11).

Fig. 4.1 shows the computational time requirements for different values of I and density for problems with number of variables = 25 and with only on diagonal elements negative.

We make the following observations from the results shown in Tables (4.2.3) to (4.2.11) and Fig. 4.1.

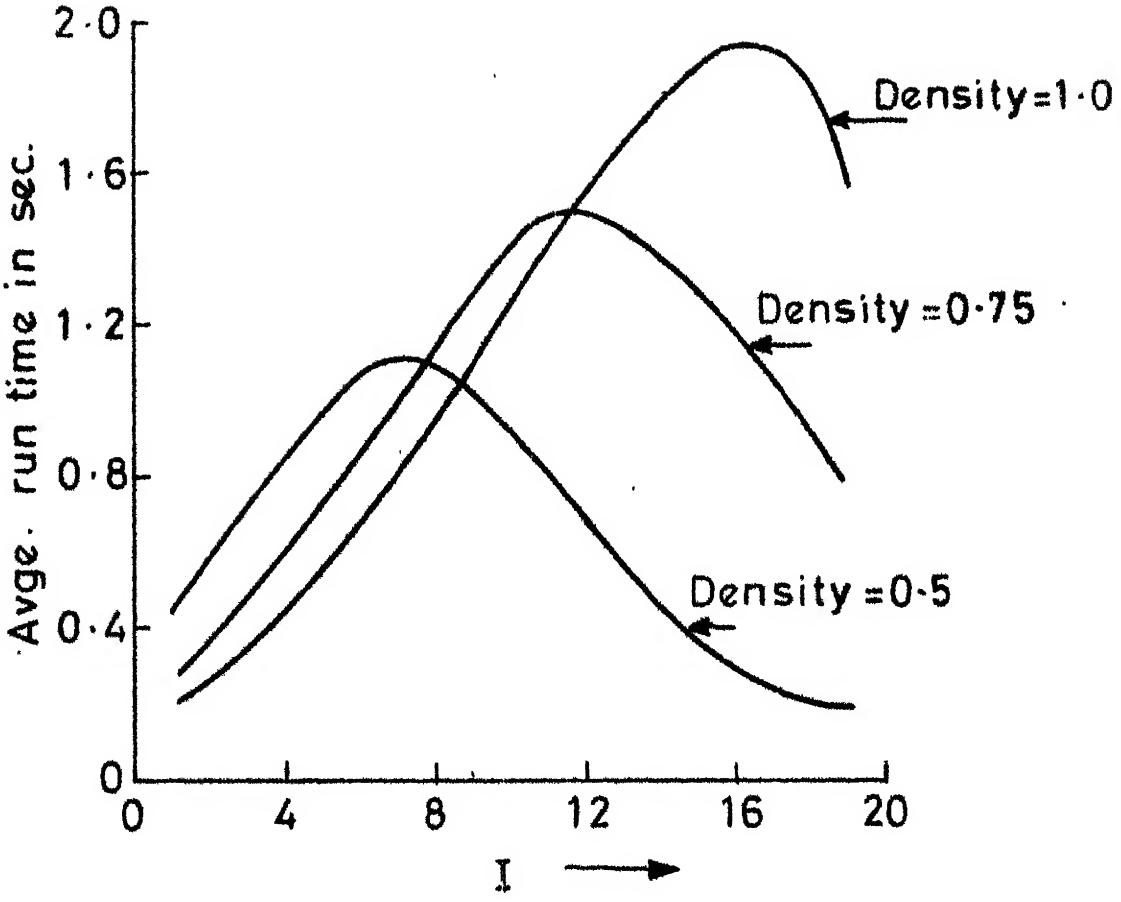

2·0
1·6
1·2
0·8
0·4
0
Avge. run time in sec.
Density=1·0
Density =0·75
Density =0·5
0
4
8
12
16
20
I

Fig. 4·1

i) The computational time required for heuristic algorithm (BEST LOCAL) is approximately constant for a given n irrespective of the factors I, IN, Density. This excludes the cases where the problem size can be reduced with trivial test considerably.

ii) Range of average percentage error is (0, 2.81).

From Tables (4.2.3)to (4.2.6) and Fig. 4.1, we make the following observations for the problems with only on diagonal elements of Q negative.

i) Computational time for branch and bound method increases with increase in value of I and then starts decreasing. The maximum is reached for values of I approximately equal to

0.75 x (n x density)

This may be due to the fact that the effectiveness of bounding test and pruning test is dependent on the negative elements of QDIAG corresponding to free variables. As the value of I increases the sum of negative elements of QDIAG corresponding to free variables also increases, hence bounding test becomes less effective for higher values of I. But, for higher values of I, since the on diagonal elements of Q and hence QDIAG are large, the trivial test and pruning test become more effective. For intermediate value of I, tests are not very effective.

ii) For low values of I computational time for branch and bound method decreases with increase in density and for high value of I the time increases with increase in density.

This may be due to the fact that the effectiveness of bounding test and pruning test is dependent on density of Q and value of I as given below:

For higher values of density when we fix a variable at 1, large number of positive elements will get added to QDIAG. Hence the sum of the negative elements of QDIAG decreases with increase in density. Hence the bounding test becomes more effective with increase in density of Q.

Since the sum of positive elements in a row of Q corresponding to free variables increases with increase in density of Q, the pruning test becomes less effective with increase in density of Q.

Also since the bounding test is more effective for low values of I, we expect the computational time required to decrease with increase in density for low values of I. Similarly, since the pruning test is more effective for high values of I, we expect the computational time to increase with increase in density for high values of I.

From Tables (4.2.8) to (4.2.11) we make the following observations for the problems with negative elements distributed throughout Q.

i) Computational time for branch and bound method increases with increase in density of matrix Q.

This is because, the sum of positive and, negative elements of Q corresponding to free variables increases with increase in density of matrix Q, hence both pruning and bounding tests become less effective with increase in density.

ii) Computational time for branch and bound is low for low and high values of percentage of negative elements in Q.

This may be, due to the fact that the trivial test is effective for low and high percentage of negative elements. Also bounding test is effective for low percentage of negatives.

iii) Computational time for branch and bound decreases with increase in I.

This may be due to the fact that the trivial test is more effective for higher values of I.

4.2.3.1 Difficult Problems:

From the above observations we conclude that degree of computational difficulty with respect to branch and bound for the following problems is maximum.

i) For problems with only on diagonal elements of Q negative:

Problems with density of matrix Q = 100 percent

$$\text{Value of I} = \frac{3}{4}\left(\frac{n \times \text{density}}{}\right) \text{(approximately)}$$

ii) For problems with negative elements distributed throughout Q:

Problems with density of matrix Q = 100 percent

Value of I = 1

Percentage of negatives IN = 50 .

4.2.3.2 Performance of Branch and Bound and Heuristic BEST LOCAL on Large Size Problems:

Table (4.2.7) shows the performance of branch and bound algorithm and heuristic BEST LOCAL for large size problems. We are able to solve 170 variable problems in about 62.13 seconds of run time by branch and bound algorithm. Heuristic BEST LOCAL solved the same problems in about 4.12 sec. Further, it is observed that as the problem size increases the heuristic solutions are deviating more from the optimal solution, although average error is within 10 percent of the optimal value.

Our branch and bound algorithm performs better than Gulati's[5] branch and pruning algorithm. For similar size problems the time with our algorithm is 1/4th to 1/5th of the time taken by Gulati's algorithm.

Table 4.2.1

Comparison of rules for selecting variable in algorithm LOCAL

L1: Select the highest negative element of QDIAG

L2: Select the first negative element of QDIAG

L3: LRC Rule

Number of problems solved for each n = 30

Rule	Number of variable n			
	20		30	
	A	B	A	B
L1	20	~~109~~ 3.6	17	~~151~~ 5.0
L2	18	~~154~~ 5.1	15	~~225~~ 7.5
L3	14	~~126~~ 4.2	13	~~210~~ 7.0

Entries in Column A show the number of problems, for which the objective function value obtained by using a particular rule is better than or atleast as good as the remaining two rules.

Entries in Column B show the ~~total~~ avge. number of iterations required to find local solution, when a particular rule is used.

Comparison between different starting point strategies.

S1: All 0's, S2: All 1's, S3: Random start with K = 9

S4: Constructive method with K = 9,

S5: Epsilon method with value of the parameter = 0.2

Number of problems solved for each n = 27

Strategy	Only on diagonal elements of Q are negative						Negative elements are distributed throughout Q					
	n = 50			n = 100			n = 50			n = 100		
	C	D	E	C	D	E	C	D	E	C	D	E
S1	21	22	30.99	24	24	38.70	13	15	94.75	8	18	13.97
S2	17	19	44.27	18	19	100.00	12	18	23.77	9	27	6.85
S3	26	27	7.63	25	26	12.73	26	27	00.69	20	27	2.47
S4	21	23	30.90	24	24	39.74	24	27	4.50	12	27	1.10
S5	19	20	30.40	21	21	37.67	12	21	20.17	2	27	4.14

C: Number of problems for which the objective function value obtained by using a particular strategy is better than or atleast as good as the remaining strategies.

D: Number of problems in which solution is within 10 percent of the best solution.

E: The highest percentage error, when a particular method is used.

Computational results for branch and bound and heuristic BEST LOCAL

Number of variables: 25

Only on diagonal elements of Q are negative

Number of problems solved for each case : 5

I	Density of matrix Q: 25 Percentage				50 percentage			
	Average run time in sec.		Average percent error	D	Average run time in sec.		Average percent error	D
	Branch and bound	BEST LOCAL			Branch and bound	BEST LOCAL		
1	.7138	.0984	1.68	2	.4516	.1058	1.19	2
3	.7240	.1064	0.00	0	.5362	.1102	0.26	1
5	.6450	.0860	0.00	0	.9904	.1056	0.18	1
7	.2900	.0550	0.00	0	1.1186	.1142	0.00	0
9	.2500	.0292	0.00	0	1.0666	.1140	0.00	0
11	.2244	.0294	0.00	0	.8592	.0998	0.00	0
13	.1796	.0156	0.00	0	.5708	.0810	0.00	0
15	.1784	.0098	0.00	0	.3520	.0448	0.00	0
17	.1784	.0078	0.00	0	.2256	.0284	0.00	0
19	.1754	.0062	0.00	0	.2028	.0184	0.00	0

D: Number of problems for which heuristic has not given optimal solution.

Computational results for branch and bound and heuristic BEST LOCAL

Number of variables : 25

Only on diagonal elements of Q are negative

Number of problems solved for each case : 5

I	Density of matrix Q: 75 percent — Average run time in sec.		Average percent error	D	100 percent — Average run time in sec.		Average percent error	D
	Branch and bound	BEST LOCAL			Branch and bound	BEST LOCAL		
1	0.2948	0.1030	1.47	2	0.2272	0.1064	0.73	2
3	0.4050	0.1084	0.00	0	0.3522	0.1092	0.63	1
5	0.6208	0.1060	0.00	0	0.5544	0.1104	0.00	0
7	0.9434	0.1124	0.00	0	0.8204	0.1126	0.00	0
9	1.3030	0.1090	0.00	0	1.1716	0.1074	0.00	0
11	1.5038	0.1058	0.00	0	1.4218	0.1118	0.00	0
13	1.4722	0.1044	0.00	0	1.6712	0.1096	0.00	0
15	1.3060	0.1068	0.00	0	1.9058	0.1126	0.00	0
17	1.0162	0.0996	0.00	0	1.9394	0.1092	0.00	0
19	0.8152	0.0952	0.00	0	1.5900	0.1138	0.00	0

D : Number of problems for which heuristic has not given optimal solution.

Table 4.2.5

Computational results for branch and bound and BEST LOCAL

Number of variables : 30

Only on-diagonal elements of Q are negative

Number of problems solved for each case : 5

I	Density of matrix Q: 25 percent — Average run time in sec.		Average percent error	D	50 percent — Average run time in sec.		Average percent error	D
	Branch and bound	BEST LOCAL			Branch and bound	BEST LOCAL		
1.	0.8960	0.1405	0.00	0	0.5630	0.1510	1.23	2
3.	0.9025	0.1490	0.00	0	1.0650	0.1495	0.00	0
5.	1.0650	0.1320	0.00	0	2.0095	0.1375	0.00	0
7.	0.9155	0.1000	0.00	0	2.7550	0.1330	0.00	0
9.	0.4600	0.0690	0.00	0	4.0665	0.1555	0.00	0
11.	0.3775	0.0580	0.00	0	4.5735	0.1395	0.00	0
13.	0.3610	0.0450	0.00	0	3.2105	0.1405	0.00	0
15.	0.2815	0.0090	0.00	0	1.4250	0.1230	0.00	0
17.	0.2850	0.0055	0.00	0	0.9240	0.0980	0.00	0
19.	0.3005	0.0095	0.00	0	0.5605	0.0865	0.00	0

D : Number of problems for which heuristic has not given optimal solution.

Table 4.2.6

Computational results for branch and bound and BEST LOCAL

Number of variables : 30

Only on diagonal elements of Q are negative

Number of problems solved for each case : 5

I	Density of matrix Q: 75 percent — Average run time in sec.		Average percent error	D	100 percent — Average run time in sec.		Average percent error	D
	Branch and bound	BEST LOCAL			Branch and bound	BEST LOCAL		
1	0.4480	0.1440	1.59	2	0.3140	0.1445	2.12	2
3	0.6760	0.1405	0.00	0	0.5590	0.1425	0.39	1
5	1.1220	0.1505	0.00	0	0.8800	0.1415	0.00	0
7	1.5850	0.1485	0.00	0	1.3930	0.1420	0.00	0
9	2.2500	0.1495	0.00	0	1.8280	0.1405	0.00	0
11	3.0950	0.1525	0.00	0	2.5410	0.1625	0.00	0
13	3.9200	0.1415	0.00	0	3.3300	0.1400	0.00	0
15	4.9290	0.1555	0.00	0	4.1750	0.1525	0.00	0
17	5.5590	0.1520	0.00	0	4.7640	0.1570	0.00	0
19	5.3570	0.1440	0.00	0	5.4120	0.1480	0.00	0
21	4.1970	0.1355	0.00	0	5.6280	0.1530	0.00	0
23	2.1920	0.1485	0.00	0	5.9920	0.1495	0.00	0
25	0.8610	0.1120	0.00	0	5.5940	0.1425	0.00	0
27	0.4570	0.0795	0.00	0	4.9180	0.1250	0.00	0
29	0.3090	0.0265	0.00	0	3.7950	0.1150	0.00	0

D : Number of problems for which heuristic has not given optimal solution.

Table 4.2.7

Computational results for branch and bound and heuristic BEST LOCAL

Only on diagonal elements of Q are negative .

Value of I : 1

Density of matrix Q : 100 percent

Number of problems solved for each case = 2

Number of variables n	Average run time in sec.		Average percent error	D
	Branch and bound	BEST LOCAL		
20	0.08900	0.0580	0.00	0
40	0.4690	0.2590	1.43	1
60	1.3570	0.5710	0.24	1
80	4.0790	1.0240	6.13	2
100	7.9750	1.5390	4.13	2
120	16.7470	2.2190	3.61	2
140	33.6890	3.0830	6.54	2
150	37.4210	3.5590	4.87	2
160	47.9600	3.7230	6.17	2
170	62.1320	4.1230	8.23	2

D : Number of problems, for which heuristic has not given optimal solution.

Table 4.2.8

Computational results for branch and bound and BEST LOCAL

Number of variables : 25

Negative elements are randomly distributed in Q

Number of problems solved in each case : 5

I	IN	Density of matrix Q: 25 percent — Run time in sec. (Average)		Average percent error	D	50 percent — Average run time in sec.		Average percent error	D
		Branch and bound	BEST LOCAL			Branch and bound	BEST LOCAL		
1	10	0.0924	0.0720	0.00	0	0.1568	0.0732	1.44	1
	30	0.3150	0.0818	0.00	0	0.7944	0.0976	0.00	0
	50	0.2948	0.0924	0.00	0	0.9530	0.1074	0.00	0
	70	0.1956	0.0306	0.00	0	0.2452	0.0824	0.00	0
6	10	0.1074	0.0334	0.00	0	0.1176	0.0364	0.00	0
	30	0.1512	0.0540	0.00	0	0.1730	0.0654	0.00	0
	50	0.1820	0.0564	0.00	0	0.3054	0.0722	0.00	0
	70	0.1204	0.0182	0.00	0	0.2138	0.0162	0.00	0
11	10	0.0998	0.0522	0.00	0	0.1114	0.0134	0.00	0
	30	0.1370	0.0450	0.00	0	0.1346	0.0480	0.00	0
	50	0.2000	0.0540	0.00	0	0.2272	0.0488	0.00	0
	70	0.1550	0.0202	0.00	0	0.1820	0.0132	0.00	0

D : Number of problems, for which heuristic has not given optimal solution.
IN : Percentage of negative elements in total non zero elements of Q.

Table 4.2.9

Computational results for branch and bound and BEST LOCAL

Number of variables : 25

Negative elements are randomly distributed in Q

Number of problems solved in each case: 5

I	IN	Density of matrix Q: 75 percent — Average run time in sec.		Average percent error	D	Density of matrix Q: 100 percent — Average run time in sec.		Average percent error	D
		Branch and bound	BEST LOCAL			Branch and bound	BEST LOCAL		
1	10	0.1344	0.0696	0.00	0	0.1500	0.0664	0.31	1
	30	1.0162	0.0990	0.16	1	1.6882	0.0926	0.14	1
	50	4.4372	0.1070	0.00	0	4.9998	0.1280	0.00	0
	70	0.2526	0.1082	0.00	0	0.2588	0.1096	0.00	0
6	10	0.1288	0.0230	0.00	0	0.1248	0.0176	0.00	0
	30	0.2812	0.0876	0.00	0	0.2970	0.0980	0.00	0
	50	0.8174	0.1054	0.00	0	2.7666	0.1014	0.07	1
	70	0.1930	0.0210	0.00	0	0.2172	0.0290	0.00	0
11	10	0.1064	0.0094	0.00	0	0.1810	0.0220	0.00	0
	30	0.1378	0.0332	0.00	0	0.1390	0.0344	0.00	0
	50	0.2480	0.0482	0.00	0	0.3232	0.0764	0.00	0
	70	0.1896	0.0210	0.00	0	0.1960	0.0162	0.00	0

D : Number of problems, for which heuristic has not given optimal solution.

IN : Percentage of negative elements in total non-zero elements of Q.

Table 4.2.1(

Computational results for branch and bound and BEST LOCAL

Number of variables : 3(

Negative elements are randomly distributed in Q

Number of problems solved in each case : 5

I	IN	Density of matrix Q: 25 percent — Average run time in sec.		Average percent error	D	5(percent — Average run time in sec.		Average percent error	D
		Branch and bound	BEST LOCAL			Branch and bound	BEST LOCAL		
1	1(	C.1725	0.1(6C	2.81	2	0.224(	C.1115	C.C(	C
	3C	0.5850	0.1415	C.CC	C	2.7(85	C.1235	C.0(	0
	5C	C.5335	0.1595	C.CC	C	4.3675	C.14CC	1.44	1
	7C	C.2725	C.C820	C.CC	C	C.5C5C	C.149C	C.CC	C
6	1C	C.193C	0.0375	0.CC	C	0.1965	0.0715	0.00	C
	3C	C.3280	0.106C	C.C0	C	1.777C	0.1340	C.CC	C
	5C	C.5CC5	0.C720	C.C0	C	0.8465	C.1305	C.00	C
	7C	C.2985	0.0310	0.0C	C	0.3035	0.1005	C.C0	C
11	1C	C.231C	0.0505	0.00	C	0.1715	0.C75C	C.C0	C
	3C	0.2515	0.0640	C.CC	C	0.2455	C.1045	0.35	1
	5C	C.2545	0.0715	C.CC	C	0.3865	0.0595	C.CC	C
	7C	C.2875	0.021C	C.CC	C	0.3CC0	0.0220	C.CC	C

D : Number of problems, for which heuristic has not given optimal solution.

IN : Percentage of negative elements in total non zero elements of Q.

Table 4.2.11

Computational results for branch and bound and BEST LOCAL

Number of variables : 30

Negative elements are randomly distributed in Q

Number of problems solved in each case : 5

I	IN	Density of matrix Q: 75 percent — Average run time in sec.		Average percent error	D	100 percent — Average run time in sec.		Average percent error	D
		Branch and bound	BEST LOCAL			Branch and bound	BEST LOCAL		
1	10	0.2395	0.1025	0.00	0	0.2980	0.1515	2.56	1
	30	5.5380	0.1395	0.00	0	10.3820	0.1470	0.00	0
	50	10.8480	0.1560	0.00	0	20.8840	0.1535	0.00	0
	70	0.6170	0.1435	0.00	0	1.2140	0.1435	0.00	0
6	10	0.1940	0.0505	0.00	0	0.2275	0.0205	0.00	0
	30	2.7980	0.1175	0.00	0	3.4015	0.1580	0.00	0
	50	3.0500	0.1400	0.00	0	4.5000	0.1485	0.00	0
	70	0.3165	0.0230	0.00	0	0.3265	0.0660	0.00	0
11	10	0.2350	0.0100	0.00	0	0.2460	0.0075	0.00	0
	30	0.2915	0.0405	0.00	0	0.3955	0.0680	0.00	0
	50	0.4765	0.1210	0.00	0	0.5365	0.1100	0.00	0
	70	0.2990	0.0180	0.00	0	0.3050	0.0120	0.00	0

D : Number of problems for which heuristic has not given optimal solution.

IN : Percentage of negative elements in total non-zero elements of Q.

Table 4.2.12

Computational results for quadratic set partitioning problems

Density of matrix Q = 100 percent

Percentage of negative elements = 0

Value of I = 1

Density of constraints' matrix A = 50 percent

Number of constraints	Average run time in sec.	
	n = 50	n = 75
10	1.772	7.879
20	1.521	6.831
30	1.412	5.674
40	1.248	4.532

4.2.4 Quadratic Set Partitioning Problems:

We solve some quadratic set partitioning problems using proposed branch and bound algorithm. The problems are first converted to equivalent unconstrained bivalent quadratic programming problems using Hammer and Rudeanu's [7] result.

For computational study, problems of size n = 50 and 75, with density of Q equal to 100 percent, value of I equal to 1, and percentage of negative elements equal to 0 are considered. The constraints' matrix A is generated using uniformly distributed numbers and the density of matrix A is fixed at 50 percent. Problems are solved for different number of constraints as (10, 20, 30, 40).

Results are reported in Table (4.2.12). It is observed that the computational time is decreasing with increase in number of constraints. Further, we have been able to solve problems with 75 variables and 10 constraints within 7.87 secs.

4.3 Conclusions:

We considered quadratic bivalent programming problem with linear constraints. Using Hammer and Rudeanu's [7] result, it is shown that the problem can be converted to equivalent unconstrained quadratic bivalent programming problem.

We have developed an algorithm LOCAL to find locally minimizing point. Different starting point strategies and, rules for selecting variable in algorithm LOCAL are considered. Computational study is performed to identify the best starting point strategy and rule for selecting variable in algorithm LOCAL. It is observed that the selection rule that selects the highest negative element of QDIAG dominates other rules. Also for starting point strategy, it is found that generating K (prefixed) random start points with varying cardinality of 1's (prefixed) is the best strategy.

Based on the above observations we suggest the following heuristic (BEST LOCAL) for getting a good local solution. i) Select K random start points with cardinality of 1's being prefixed. ii) For each of the starting point get a local solution using algorithm LOCAL. iii) Select the best local solution.

For global minimization, a branch and bound algorithm is developed for which the initial incumbent is made equal to the objective function value from heuristic BEST LOCAL. Effective bounds are developed for pruning. In addition, Gulati's [5] pruning test is also used for pruning.

Computations are performed, for both branch and bound and heuristic BEST LOCAL, on two types of problems: 1) The problems with only on-diagonal elements of Q negative,

2) The problems with negative elements distributed throughout Q. The effect of different parameters namely, i) Density of matrix Q, ii) I : Ratio of size of on diagonal to off-diagonal elements of Q, iii) Percentage of negative elements in total non-zero elements of Q, on the performance of the branch and bound, heuristic BEST LOCAL is studied.

Heuristic BEST LOCAL is compared with branch and bound algorithm in terms of computational time and average percentage of error, for different types of problems and different combinations of the parameters. It is observed that the average computational time required for heuristic BEST LOCAL is approximately same for given n, irrespective of values of other parameters. Also average percentage of error for the heuristic BEST LOCAL is varying from 0 to 2.81.

Computational performance of branch and bound algorithm is encouraging. We have been successful in solving problems with 170 variables, (only on diagonal elements of Q being negative, density of Q equal to 100 percent and value of I equal to 1), within 62 seconds of computational time. Earlier Gulati [5] has reported solving 125 variable problem within 198 seconds on same computer.

We further observe that for computational purpose following problems are difficult to solve and can be used as test problems for the future studies.

1. Density of Q = 100 percent, ratio of on diagonal to off diagonal elements' size = 1, percentage of negative elements = 50, negative elements distributed throughout Q.

2. For problems with only on diagonal elements negative, density of Q = 100 percent, ratio of size of on diagonal elements to the off diagonal elements = $\frac{3}{4}$ (density x no.of variables).

Further, we have been able to solve quadratic set partitioning problems with 75 variables and 10 constraints (density 50 percent of the constraint matrix, 100 percent Q, I = 1) within 7.87 seconds. We also observed that as the number of constraints in the set partitioning problem increases, the time to solve the problem decreases.

A description of the computer program is given in Appendix I.

REFERENCES

1. Balas, E., "An additive algorithm for solving zero-one linear programs," Operations Research 13 (1965), 517-546.

2. Francis, R.L., and White, J.A., "Facility layout and location," (Prentice-Hall, Englewood Cliff, N.J.) 1974).

3. Gallo, G., Hammer, P.L. and Simeone, B., "Quadratic knapsack problems," Research Report, 76-43, Univ. of Waterloo, Waterloo, Canada (1974).

4. Geoffrion, A.M., and Graves, G.W., "Scheduling parallel production lines with change over costs, practical application of quadratic assignment/LP approach," Operations Research 24 (1976), 595-610.

5. Gulati, V.P., "On quadratic bivalent programming problems", Ph.D. Thesis submitted at I.I.T. Kanpur, Dec. 1980.

6. Gulati, V.P., Mittal, A.K. and Gupta, S.K., "Minimising unconstrained quadratic bivalent functions", Paper submitted to European Journal of Operations Research.

7. Hammer, P.L., and Rudeanu, S., "Boolean methods in operations research and related areas", Springer-Verlag, Berlin-Heidelberg - New York, (1968).

8. Hammer, P.L., and Hansen, P., "Quadratic 0-1 programming," Core discussion paper 7219, April, 1972.

9. Hammer, P.L. and Peled, U.N., "On the maximization of pseudo-boolean function", JACM 19(1972), 165-282.

10. Hammer, P.L., "Boolean procedure for bivalent programming", in Hammer and Zountedijk (Eds.) Mathematical programming and applications, North-Holland/ America Elsevier (1972).

11. Hansen, P., "Methods of non-linear 0-1 programming", Annals of discrete mathematics, 5 (1979), 53-70.

12. Hillier, F.S., "The evaluation of risky interrelated investments", North Holland Publishing Company, Amsterdam and London (1969).

13. Knarup, J., and Fruzen, P.A., "Computer aided layout design," Mathematical programming study 9(1978), 75-94.

14. Laughhaunn, D.J., "Quadratic binary programming with applications to capital-budgeting problems," Operations Research, 14 (1970), 454-461.

15. Laughhaunn, D.J., "Capital expenditure programming and some alternative approaches to risk," Management Science 17 (1971), 320-336.

16. Lawler, E.L., "The quadratic assignment problem: A brief review" in B. Roy, ed., Combinatorial Programming: Methods and Applications (1975).

17. Markowitz, H.M., "Portfolio selection," Monograph 16 Cowels foundation, Wiley, New York (1959).

18. Picard, J.C. and Ratliff, H.D., "A graph-theoretic equivalence for integer programs," Operations Research 21 (1973), 361-369.

19. Picard, J.C. and Ratliff, H.D., "Minimum cuts and related problems," Networks 5 (1975), 357-370.

20. Picard, J.C. and Ratliff, H.D., "A cut approach to the rectilinear facility location problem," Operations Research 26 (1978), 422 - 433.

21. Robin Segerblom Liggett, "The quadratic assignment problem, An experimental evaluation of solution strategies," Management Science, 27 (1981), 442-458.

22. Rosenberg, I., "Reduction of bivalent maximization to quadratic case," Cabiers Centre Etudes Rech. Oper., 17 (1975).

23. Rosenberg, I., "0-1 Optimization and non-linear programming," Rev. Francise Informat. Recherche Operationnelle 6 (1972), 95-97.

24. Rosenberg, I., "Minimization of pseudo-boolean function by binary development," Discrete Mathematics 7 (1974), 151-165.

25. Taha, H., "A Balasian-based algorithm for zero-one polynomial programming," Management Science 18 (1972), B 328-343.

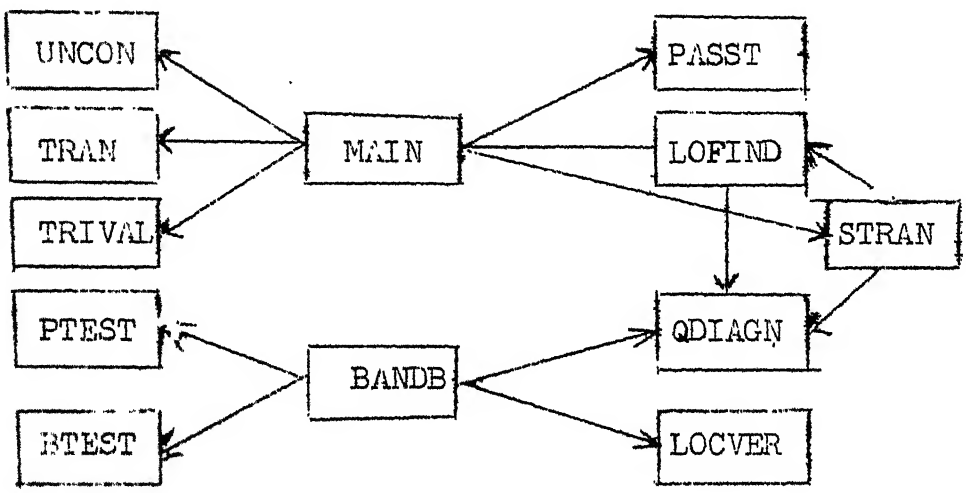
26. Taha, H., "Further improvements in polynomial zero-one algorithm," Management Science, 19 (1972), 226-227.

27. Watters, L.J., "Reduction of integer polynomial programming problems to zero-one linear programming problems," Operations Research 15 (1967), 1171-1174.

APPENDIX I

DESCRIPTION OF MODULES FOR BRANCH AND BOUND AND HEURISTIC BEST LOCAL

The linkages of different modules are shown below:



Description of the Modules:

1. MAIN: This is the main program, from which we call other programs depending on our requirements.

We give format free input to the program as given below:

Card 1: Number of variables, Code 1, Code 2, K

where Code 1 = 1 if exact solution is required

else 0

Code 2 = 1 if the constraints are present

else 0

K = Number of starting points to be used in heuristic BEST LOCAL.

Card 2 onwards: Punch elements of matrix Q row wise. Then punch elements of constraints' matrix A row wise and finally the elements of vector b.

The output will be as given below:

Optimal objective function value and a vector containing 0 and 1's denoting optimal solution X.

Subroutines called: UNCON, TRAN, TRIVAL, PASST, STRAN, LOFIND.

2. BAIDB: This is the program for branch and bound algorithm.
   Subroutines called: PTEST, BTEST, QDIAGN, LOCVER.

3. STRAN: This is the program for heuristic BEST LOCAL.
   Subroutines called: LOFIND, QDIAGN.

4. LOFIND: This is the program for algorithm LOCAL.
   Subroutines called: QDIAGN.

5. QDIAGN: This is used for finding the values of QDIAG (X^k) from QDIAG (X).

6. LOCVER: This is used to verify, whether a particular node is type I or II.

7. PASST: This calculates the sum of positive and negative elements of rows of Q, for different levels of fixing of variables.

8. PTEST: This is the program for pruning test.
9. BTEST: This is the program for bounding test.
10. TRIVAL: This is the program for trivial test.
11. TRAN: This program rearranges the elements of Q in ascending order of the sums of negative elements of rows of Q.
12. UNCON: This program converts the constrained quadratic bivalent programing problem to equivalent unconstrained bivalent quadratic programing problem.